METHOD FOR RESOURCE MANAGEMENT IN A
REAL-TIME EMBEDDED SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

5   Not applicable.

STATEMENT REGARDING FEDERAL SPONSORED RESEARCH OR
DEVELOMENT

Not applicable.

10

FIELD OF THE INVENTION

The present invention relates to the allocation and usage of processor resources in the performance of processing tasks that have time-varying changes in resource requirements. More specifically, the present invention concerns monitoring the

15   processor resources and determining or estimating the current needs for these resources so that they may be allocated by the processor in an optimally efficient way, no matter what kind of event may happen in the future.

BACKGROUND OF THE INVENTION

20   A software developer has a limited number of resources to allocate to a processor for use in performing one or more executable functions. These resources may include the memory, processing speed, millions of instructions per second (MIPS), processing time, etc., that can be allocated to one or more functions or multiple states of a function. Because of the limited processor resources, a programmer must attempt to write

25   programming that most efficiently utilizes the resources of the processor.

Another concern for the programmer is the dynamically varying usage of the

resources over time. In a real-time embedded system, the signal input characteristics determine which functions will run. Therefore, resource consumption depends on the signal input. Also, adaptive algorithms change the mode of task execution in accordance with the signal environment and the achieved performance, thereby changing the amount

5   of resource consumption. Unfortunately, programmers do not have the benefit of real-time information indicating the dynamic usage of processor resources, when designing and implementing a program function. For example, determining the dynamic utilization of the MIPS resource by a previously known method requires that the software function toggle an output pin of the processor each time the function begins and finishes.

10

Existing methods for minimizing a processor's performance degradation include time slicing and background processing. For example, when the available memory capacity of a digital signal processor (DSP) is nearly used up or overloaded, processing operations become prioritized. Prioritizing the operations allows those having a high

15   priority to be performed in the foreground and lower priority operations performed in the background. Channels are allocated MIPS for calculations whether the channel uses the MIPS or not.

SUMMARY OF THE INVENTION

20   The invention relates to a resource management agent (Agent) used to manage resources in a processor. This Agent serves to monitor, determine, and control resource consumption. Real-time resource management within a processor allows far more tasks to be performed in a particular time period. For example, such resource management used with a communication processor may increase the number of communication

25   channels that may be supported simultaneously by a digital signal processor.

The resource management Agent controls the allocation of processing resources

TI-32882

assigned to discrete parts of a decomposed algorithm, when these parts are capable of being managed (i.e., turned on and off) by the Agent. In other words, the Agent dynamically reassigns processing resources so that they are efficiently used to satisfy the time-varying requirements of the decomposed algorithm parts. Resources are assigned to parts of the algorithm as they are needed. The amount of resource used by a part of the algorithm is estimated by the Agent, based on the current mode of execution.

A preferred embodiment of the above-described invention relates to a method of allocating the processor MIPS to tasks in a queue 20 waiting to be executed. This method includes the steps of:

(1)   determining the number of processor MIPS available to be assigned;

(2)   determining an estimate of the number of MIPS needed for each task waiting in the queue to execute at a maximum intended speed for the processor; and

(3)   allocating the available MIPS to the tasks based on a hierarchical priority scheme, where the priorities are based on environmental conditions and the achieved performance.

To control peak MIPS consumption, the Agent stores an estimate of peak MIPS usage by specific software functions locally and updates the estimate whenever the state of the function changes. The estimates are subsequently used in a queuing scheme to determine how many and which of the executing software instances may enable the functions available to them, without exceeding a maximum resource threshold. When an algorithm is broken into separate parts and the parts are manageable such that they can be turned on and off, the Agent controls the way processing resources are used by the algorithm. Processor resources are applied where they are needed and are most effectively used.

Prior to managing processing resources, the agent determines the resource usage of each part of an algorithm. Based on internal information passed from the algorithm to the Agent, external resource allocation limits of the software and processor design, environmental conditions, and achieved performance, the Agent distributes processing

5 resources to the parts of an algorithm that have the greatest need while taking resources from parts that can operate with less resource allocation or no allocation at all. As opposed to allocating a certain amount of resources to certain tasks, the Agent is dynamic and can reallocate processing resources to parts of algorithms as they need more processing power and reduce the allocation when the processing can be reduced.

10

The Agent has alarms set at high and low resource usage thresholds. When the processor's resource is running low or completely allocated and another part of the algorithm requires the resource, the Agent analyzes the subroutines within the algorithm and the input channels to prioritize the allocation of the resource among the competing

15 algorithm parts, based on the environmental conditions and achieved performance. Lower prioritized resource allocations are redirected to the parts of the algorithm that have greater priority. Even if all channels of a processor require a large allocation of the resource simultaneously, the Agent limits the consumption of the resource through graceful degradation of performance. The degradation does not cause the processor to

20 lose information or cause the processor to crash. Some compromise in software performance may occur to the user but is corrected as the Agent frees and reallocates the resource on a dynamic basis.

The Agent is similar to a flow control. It directs more resource to modules and

25 channels that have the most instant resource needs while removing the resource from those modules that have an over-allocation of the resource. The Agent can dynamically update scheduling priorities based on various performance measures for the algorithms it

controls. The Agent uses both internal and external controls. Each module contains an estimate of its resource needs and supports the ability to have its resource consumption reduced by the processor. The external controls slow down all processing or perform performance-degrading reallocation of resources when a greater amount of the resource is needed by an algorithm than is available at that time.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the invention are discussed hereinafter in reference to the drawings, in which:

Figure 1 -   illustrates a set of software processes operating for the corresponding set of active instances;

Figure 2 -   illustrates a communication processor interfaced with a plurality of communication channels through its communication ports; and

Figure 3 -   illustrates a representative round robin allocation of a resource to the functions of four concurrently executing instances.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Referring now to Figure 2, a communication processor 20 is interfaced with a plurality of communication channels 22 through its communication ports 21. Each of the communication channels 22 is capable of conveying an analog signal between the communication processor 20 and a channel terminating device. Upon receipt of an analog signal, the communication processor 20 creates a digital representation of the analog signal using numerous digital signal processing functions. Each channel port 21 is continuously monitored by the communication processor 20 to determine when a channel link has been established or extinguished on the communication channels 22. While a channel link exists, the communication processor 20 assigns its resources to functions that

TI-32882

digitize and improve the digital representation of the analog signal. The assigned resources may be MIPS, memory, and other resources of the communication processor 20.

5   Referring now to Figure 1, a software process 1-3 is executed by the communication processor 20 for each instance of a channel link. The instance is created when the link is established and extinguished when the link terminates. Figure 1 illustrates a set of software processes 1-3 operating for the corresponding set of active instances identified by the instance index pointer j. The instances illustrated are those

10   identified by the instance index values of $j = \{1, 2, \ldots, m\}$.

        Each software process 1-3 operates in the same manner. Therefore, for simplicity, the invention will be described in reference to only one instance of the software process and this description will apply equally well to every other instance of the software

15   process. Moreover, each software process 1-3 completes its execution within a period of time t. Though the software process 1 is completed in the time period t, it is serially repeated for each incremental period of time that the instance remains active. The time period t is minimally bounded the amount of time required to completely execute any one of the processing functions operating on the channel link instance. It may be a uniform or

20   varying period, but is assumed to be a uniform period for the purpose of describing the invention. The processing functions are discrete parts of a decomposed algorithm that may be executed independently of other parts of the algorithm.

        After the software process 1 begins, as indicated by reference numeral 4, two

25   index pointers, j and k, are initialized 5. The instance index pointer j is set to point to the next unused instance value available in the instance index. A function index pointer k is initialized to point to the first value in the index of processing functions that may be

TI-32882

executed by the software process in connection with the channel link instance. For the first instance of a channel link, the instance index pointer j is given a value of 1, as indicated by reference numeral 5. Similarly, the instance index pointer j is given a value of 2, as indicated by reference numeral 18, for the second instance of a channel link and a value of m for the mth instance, as indicated by reference numeral 19.

For each time period t, the communication processor 20 determines the number of instances in existence. The processor 20 makes a determination of the amount of resources that each instance needs to execute the functions that are appropriately performed on the instance in its present state. If adequate resources are available to perform the appropriate functions on every existing instance, then these resources are distributed accordingly. However, if inadequate resources are available, then the communication processor 20 must prioritize the allocation of resources to the pending functions of each instance, based on the environmental conditions and achieved performance. The allocation is implemented such that some functions of an instance may be executed and others may not. Those that are executed receive processor 20 resources for their execution. Each of the functions within the process may be assigned a separate priority within the hierarchical priority scheme. Similarly, each instance of each function may be assigned a separate priority within the hierarchical priority scheme, based on the environmental conditions and achieved performance.

The amount of a resource allocated by the processor 20 to execute the pending functions of an instance, for the current time period, may be expressed by the equation:

$$R_j = m_0 + \sum_{k=1}^{N} a_{jk} \times f_k \left(environmental\ inputs_j, achieved\ performance_j \right)$$

where,

TI-32882

R_j = the amount of a resource allocated to the jth instance;

N = the number of pending functions for the jth instance;

m_0 = the amount of a resource required to execute the background processing of the jth instance, excluding the resource allocated to the pending functions of the jth

5 instance;

f_k (environmental-inputs$_j$, achieved-performance$_j$) = the amount of a resource required to execute the kth pending function, based upon the current state of the environmental inputs and the achieved performance of the jth instance;

a_{jk} = 0, if no resource is to be allocated to the kth pending function of the jth

10 instance; and

a_{jk} = 1, if resource is to be allocated to the kth pending function of the jth instance.

The amount of resource required by the kth function, f_k, in the jth instance is variable and depends upon the state conditions of the jth channel link. The state conditions vary in

15 accordance with the environmental inputs of the channel link and its achieved performance, during the current time period t. Priorities are assigned to the pending functions based on the environmental inputs of the channels, the achieved performance of the channels, and the amount of resources recently consumed by the active channel instances. The assignment of priorities to the k pending functions of the j instances may

20 be expressed by the equations:

$$p_{jk} = g_k \left(environmental\ inputs_j, achieved\ performance_j, recently\ consumed\ resource_j \right)$$

where $0 \leq p_{jk} \leq 1$;

and,

p_{jk} = the priority assigned to the kth function of the jth instance; and

25 g_k = is a function that assigns a priority to the kth function of the jth instance based on the environmental inputs of the jth channel instance, achieved performance of the jth

TI-32882

channel instance, and the amount of resource recently consumed by the jth instance.

To achieve the prioritized implementation of a set of functions, f_k, in the jth instance, the communication processor 20 assigns a binary value of either zero or one to the a_{jk} of each kth pending function of the jth instance. Reference numeral 6 identifies the point in the process flow 1 where the value assigned to the a_{jk} associated with the first pending function of the jth instance is evaluated to determine whether this function will be executed in the current time period. If the value of a_{jk} is zero, the function will not be executed in the current time period t and the process flow 1 will continue with the next step of the process, identified by reference numeral 8. If the value of a_{jk} is one, then the first function will be executed in the current time period, as indicated by reference numeral 7 and the process flow 1 will continue with the step identified by reference numeral 8.

Next, the function index pointer is incremented by a value of one to point to the next function in the index, as indicated by reference numeral 8. Again, the process flow 1 evaluates the value assigned to a_{jk} for the kth pending function of the jth instance, as indicated by reference numeral 9. In this case, if the value of a_{jk} associated with the second pending function of the first instance is one, the second function for this instance will be executed in the current time period, as indicated by reference numeral 10. If the value of a_{jk} is zero in this instance, then the second function will not be executed in the current time period and the process flow continues at the step identified by reference numeral 11. Similarly, the process flow continues at the step identified by reference numeral 11 after the second function is executed.

Reference numerals 11-13 identify the steps of the process flow 1 where the function index pointer is incremented, the value assigned to a_{jk} for the third pending

function of the jth instance is evaluated, and this third function is executed in the current time period, if the value of a_{jk} is one for the indexed values of j and k. This process of incrementing k, evaluating a_{jk}, and executing the kth function of the jth instance, for the indexed values, is repeated until it has been applied to all of the N functions of the jth
5   instance, as indicated by reference numerals 14-16. Thereafter, the process flow 1 for the jth instance, of the current time period, is terminated, as indicated by reference numeral 17.

Referring now to Figure 3, imagine, for the purpose of describing the invention,
10   that a separate communication link is received on each of four communication ports 21 of the processor 20. Each communication link creates a separate instance for the processor 20 to execute for every period t throughout the duration of the communication link. These instances are identified as instance one 30, instance two 31, instance three 32, and instance four 33. Each instance 30-34 has two functions, f_1 34 and f_2 35, that may be
15   applied to its respective communication link. The horizontal axis of Figure 3 has been sub-divided into 7 distinct time periods t_0-t_6 36-42, respectively. For each time period, the processor 20 assigns a value of zero or one to the a_{jk} associated with the functions of each instance.

20   For the purpose of describing Figure 3, assume that each function uses a fixed amount of a particular resource and the resource of concern is the millions of instructions per second (MIPS) that a function needs to execute in an instance. Further assume that the communication processor 20 has a maximum of 100 MIPS to allocate, all of the processor MIPS may be allocated to the processing functions f_1 and f_2, and the functions
25   require the following numbers of MIPS: f_1 = 25 MIPS and f_2 = 50 MIPS. Though all four instances of the communication links need to be acted upon by the processing functions, there are insufficient MIPS for the functions f_1 34 and f_2 35 to execute on each instance

30-33, in a single time period. Therefore, a round-robin scheme may be used to apply the two functions 34 and 35 to each of the instances 30-33 equivalently. In the case of a round-robin scheme, all of the priorities p_{jk} for the pending functions are equal and remain fixed.

5

In general, the number instances to which a function may be applied is given by the equation:

$$\sum_{j=1}^{C} a_{jk} = C_{0k} \leq C$$

10 where:

C is the number of instances (i.e., communication links); and

C_{0k} is the maximum number of instances to which the k_{th} function may be applied, during a single time period t, and identifies the maximum number of slots for the k_{th} function.

15

Referring again to Figure 3, a_{11}, a_{12}, and a_{21} have been assigned a value of one by the processor 20 and all other a_{jk} for the first time period, t_0 36, have assigned a value of zero. Since each instance of function f_1 34 consumes 25 MIPS and each instance of function f_2 35 consumes 50 MIPS, the 100 MIPS available to the processor 20 have been

20 allocated. In the illustrated case, the maximum number of slots, C_{0k}, available to function f_1 34 is one and the number available to function f_2 35 is two, for each time period t.

No further prioritization of the functions f_1 34 and f_2 35, within the four instances, is provided in the example of Figure 3. The processor 20 simply provides the MIPS

25 resources to each instance in a round-robin fashion over multiple time periods t. This

11   TI-32882

may be seen by the diagonal movement of the values assigned to the a_{jk} as time progresses from t_0 to t_6. Notice the value assigned to the a_{jk} for both functions of the first instance, in time period t_0, moves progressively to the a_{jk} of the two functions assigned to the other instances with each incremental time period. The value of a_{jk} in the tabular cell position identified by reference numeral 43, in period t_0, moves through the matrix of a_{jk} in the manner tabulated in Table 1.

Table 1

Item	Period	a_{jk}	Referenced Cell
1	t_0	a_{11}	43
2	t_1	a_{21}	45
3	t_2	a_{31}	47
4	t_3	a_{41}	49
5	t_4	a_{11}	51
6	t_5	a_{21}	53
7	t_6	a_{31}	55

Similarly, the value of a_{jk} in the tabular cell position identified by reference numeral 44, in period t_0, moves through the matrix of a_{jk} in the manner tabulated in Table 2.

Table 2

Item	Period	a_{jk}	Referenced Cell
1	t_0	a_{12}	44
2	t_1	a_{22}	46
3	t_2	a_{32}	48

TI-32882

Table 2

Item	Period	a_{jk}	Referenced Cell
4	t_3	a_{42}	50
5	t_4	a_{12}	52
6	t_5	a_{22}	54
7	t_6	a_{32}	56

Although the estimated amount of a resource needed to execute a function may be known *a priori*, the actual amount of the resource needed for a particular application of the function to an instance may not be known. Recall that the amount of a resource required to execute the kth pending function is variable and is based upon the current state of the inputs and performance of the jth instance.

When estimating the amount of resource needed for the function to execute, the processor 20 bases the estimate on the maximum amount of the resource that the function can use. Often, the function uses less than the maximum amount of the resource that it is capable of consuming. To optimize the efficient use of the resource, the processor 20 will attempt to over-allocate the resource based upon the maximum consumption rate. The processor 20 then monitors the actual consumption of the resource by the function. If, collectively, the executing functions consume an amount of the resource exceeding a high threshold value, then the processor 20 begins to reduce the amount of the resource allocated. On the other hand, if the executing functions collectively consume less of the resource than the value indicated by a low threshold, the processor 20 attempts to maximize the allocation of the resource. Another way of describing this feature is in terms of a consumption alarm. If the actual consumption of the resource exceeds the high threshold value, then the consumption alarm is set and the allocation of the resource is

13

reduced. If the actual consumption of the resource falls below the low threshold value, an existing alarm condition is removed and the processor allocates resources normally.

There are two ways of reducing the amount of the resource allocated. First, the processor can reduce the number of instances during which a particular sub-set of the functions execute. Essentially, this is accomplished by reducing the queue sizes of the executing functions. The queue size identifies the number of instances of a function that may execute concurrently. A queue size may be varied between a minimum size of one and the maximum number of instances that exist. Second, the processor 20 can reduce the amount of the resource allocated to a sub-set of the executing functions. In this second way, the processor 20 reduces (i.e., throttles) the amount of the resource that an executing function may consume.

As mentioned before, the resources controlled by the processor 20 may be MIPS, memory, and other resources of the communication processor 20. Continuing with the example where the resource is the processor MIPS, a way of regulating the allocation of MIPS in response to their actual consumption is described. For some period of time, τ, a measurement is made of the processor's 20 idle durations. These idle durations are summed to generate the total idle time, t_{idle}, for the period τ. The amount of MIPS actually used by the processor 20 during this period may be derived using the equation:

$$Total\ Number\ of\ MIPS\ Used = \left(1 - \frac{t_{idle}}{\tau}\right) \times Total\ Processor\ MIPS$$

where,

total processor MIPS = the maximum number of MIPS that is achievable by the processor.

14   TI-32882

Once the processor determines the MIPS actually consumed by the totality of executing functions, it may compare this amount to the high and low threshold values. If the measured value exceeds the high threshold value, the processor 20 instructs the Agent to reduces the allocation of MIPS over all active instances and functions that are

5 considered for execution. If the measured value is less than the low threshold, then the processor 20 attempts to increase the allocation of MIPS. The process of measuring the actual MIPS, comparing the measured value to threshold values, and adjusting the allocation of MIPS as necessary is performed serially in time period and may be performed periodically or intermittently. Allocation of the available MIPS to the

10 functions waiting in the queue may be conducted to optimize the number of MIPS assigned to these functions, to optimize the number of instances of the functions concurrently being executed, or according to some other scheme.

Because many varying and different embodiments may be made within the scope

15 of the inventive concept herein taught, and because many modifications may be made in the embodiments herein detailed in accordance with the descriptive requirements of the law, it is to be understood that the details herein are to be interpreted as illustrative and not in a limiting sense.

TI-32882